AR-009-692

DSTO-GD-0081

Low Cost INS-GPS Integration for Navigation (INS Software Implementation)

N. Luckman

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Low Cost INS-GPS Integration for Navigation (INS Software Implementation)

N. Luckman

Weapons Systems Division
Aeronautical and Maritime Research Laboratory

DSTO-GD-0081

ABSTRACT

This document describes several versions of software that have been developed to implement an Inertial Navigation System algorithm. The form of the algorithm is intended to facilitate the combining of Global Positioning System data into the INS solution, and a simple example (position and velocity reset) is included in the software.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 8111
Fax: (03) 9626 8999

AR No. 009-692
March 1996

APPROVED FOR PUBLIC RELEASE

Low Cost INS-GPS Integration for Navigation (INS Software Implementation)

Executive Summary

Inertial Navigation Systems (INS) and Global Positioning Systems (GPS) both can be used for a wide range of navigation functions. Each has its strengths and weaknesses in this respect.

The major strengths of GPS are that it can provide regular estimates of position that will remain within certain error bounds and that its hardware costs are relatively cheap. Its weaknesses are: the size of the errors from GPS receivers (not requiring approval from the US Department of Defense) are relatively large (around 100 m); the update rate for navigation solutions are relatively slow (1 to 10 Hz); and the receiver output is unreliable due to received signal dropping out or being jammed.

For INS its strengths are: a high navigation solution rate; low errors over a short period of time; and reliability as it is self contained. The major weakness is that errors grow with time and the rate of growth is inversely related to the cost of the INS.

Combined INS-GPS systems are capable of making up for the weaknesses inherent in each. For example self guided missiles need a high degree of accuracy in the end game scenario which may be achieved with high cost INS guidance alone but with the incorporation of GPS much cheaper INS units could be used for the same result.

This document describes a software implementation of an INS algorithm that will allow easy integration of GPS data into the navigation solution taking into account issues such as different data rates from the inertial sensors and GPS receivers. This software will be useful for research into optimisation algorithms for INS-GPS integration.

Contents

1. Introduction

Guidance and Control Group of Weapons Systems Division has a task ***GPS-Guided Weapons Applications (ADA93/276)*** in which research is being undertaken to develop navigation techniques for self guided unmanned vehicles based on low-cost Inertial Navigation System (INS) and Global Positioning System (GPS) technologies.

2. Background

Strapdown INS technologies are based on the principle of integrating specific forces and rates measured by accelerometers and rate gyros of an Inertial Measurement Unit (IMU) fixed to the navigating body. Given the initial conditions of position, velocity and attitude, accurate real time integration of IMU output will produce position and attitude information in some given navigation coordinate system. GPS on the other hand relies on the technique of comparing signals from orbiting satellites to calculate position (and possibly attitude) at regular time intervals, but being dependent on the satellites signals makes GPS less reliable than the self contained INS due to the possibility of drop-outs or jamming.

The errors inherent with INS, ie turn-on bias, dynamic biases and integration, are relatively small over update periods, however they accumulate with time. Errors with GPS are bounded with time, however they are large relative to those of INS over an update period.

The two technologies are complimentary so that combining them provides an opportunity to overcome their respective weaknesses. Combined INS-GPS systems usually rely on high accuracy IMU's (hence expensive) while using less than optimal algorithms to combine the data. By using improved algorithms it is believed that less accurate, but relatively inexpensive IMU's could be used in combined INS-GPS systems.

3. Aims

In support of the development of algorithms to combine INS and GPS data this document describes the software implementation of an INS algorithm the structure of which will allow the incorporation of GPS data, and that of other sources, such as magnetometers, altimeters etc.

4. Considerations for Combining INS-GPS Data

The following discusses some of the specific issues peculiar to INS and GPS systems that will have a bearing on how a combined INS-GPS system may be implemented.

4.1 INS

In integrating IMU data to obtain a navigation solution, the sampling period in an integration step is important to consider. For a strapdown IMU, body specific forces and rates are likely to have some relatively high frequency components hence a high sampling and integration rate is needed to deal with these. On the other hand accounting for the motion of the navigation reference frame (usually Earth based reference) with respect to inertial space (in which an IMU operates), need only be performed at a much slower frequency since the motion of the Earth's surface in space is both smooth and slow (in rotation). Other considerations for IMU integration such as the value for gravity as a function of altitude and latitude need only be sampled at slower frequencies.

Computational time becomes an issue given that it is desirable to perform at least some of the integration process at a high rate, hence there is a need to minimise the computational over-head. An approximate solution to the equations of motion that cuts computational time is a compromise between these conflicting requirements.

4.2 GPS

GPS receivers can operate on at least one of two coded messages transmitted by the satellites. One of the messages is encrypted and is only accessible with the approval of the US Dept. of Defense. The other is open to public use however the information it contains has been deliberately corrupted so that the ultimate accuracy of receivers using it alone are an order of magnitude worse than those using the encrypted message, the latter being capable of less than 10 meters accuracy.

It is possible to counter the intentionally added errors using differential techniques where a second receiver at a known location transmits error information to the first that is common to both receivers. This is not practical for the intended purposes of the navigation system being studied so need not be considered further here.

Most GPS receivers will generate navigation solutions at a rate of 1 Hz, however some receivers will produce at least a partial solution at rates up to 10 Hz.

4.3 Combining INS and GPS

Whatever technique is chosen to fuse the data from the IMU and GPS units the major consideration arising from the previous sections is that of the rates at which data becomes available. With a GPS unit issuing data at a slower rate than an IMU unit the algorithm must be able to integrate the IMU data at the required higher rate and then periodically be able to fuse the resulting data with GPS data. This requirement means other data sources also should be easily accommodated with respect to their data rates.

5. INS Integrating Algorithm

With the above in mind the algorithm chosen for the integration of IMU data is that defined by Miller (ref 1). The following sections describe in some detail Miller's algorithm.

5.1 Algorithm Strategy

Miller's algorithm breaks the various processes up according to relative required rates of calculation. This provides a natural means of incorporating GPS data at an appropriate rate.

Three rates are used:

Fast Rate: At the fastest rate raw IMU data is adjusted to account for inherent biases then integrated to give a representation of attitude and velocity and in the inertial frame. The velocity represents only that accumulated from forces since the last iteration of calculations at the intermediate rate.

Intermediate Rate: At the intermediate rate inertial frame attitude is adjusted to allow for Earth rotation. The velocity increment since the last intermediate calculations due to inertial forces is then converted to the navigation frame. The total velocity increment in the navigation frame needs to take into account velocity changes due to gravity and corriolis effects. The velocity increment due to these effects is calculated at this rate and added to the inertial velocity increment to give the total velocity increment since the last intermediate calculations. Total velocity is then updated and intergrated to give the change in postion in the navigation frame over the last intermediate period.

Slow Rate: At the slow rate latitude, longitude, gravity, and Earth radius are updated.

The form of data available at the intermediate rate from the INS algoritm is compatible with that from the GPS. This means that it would be appropriate to perform any INS and GPS data fusion algorithm at a rate no greater than the intermediate rate.

5.2 Attitude Representation

Miller utilises quaternions as the method for representing attitude of the IMU. The following gives a basic description of the particular form of quaternions used by Miller. Appendix 1 provides some further detail on their definition and properties.

A quaternion uses four parameters to describe the rotation from one set of axes to another. In essence it consists of a three element vector that is common to both axes and a scalar value that defines the rotation about that vector.

The governing differential equation for rotation in terms of quaternions defined in Appendix 1 is

$$\dot{\overline{Q}}(t) = \tfrac{1}{2}\overline{Q}(t)*\omega(t)$$

where $\overline{Q}$ is the quaternion and ω is angular velocity (which can be represented as a quaternion with a zero scalar term), and * is the quaternion multiplication operator (see Appendix. 1).

For a system where discrete sampling is carried out a means of calculating $\overline{Q}(t+\delta t)$ is required. Miller describes several methods to obtain a solution for this including 3rd order Taylor Series and 4th order Runge-Kutta (see Appendix 2). Miller also describes the following solution.

$$\overline{Q}(t+\delta t) = \overline{Q}(t) * \overline{\theta}$$

where $\overline{\theta}$ is a rotation quaternion defined as

$$\overline{\theta} = C, \theta S \qquad \text{with } C = \cos(\tfrac{1}{2}\theta_0),\ S = (1/\theta_0)\sin(\tfrac{1}{2}\theta_0), \text{ and } {\theta_0}^2 = \theta \cdot \theta .$$

The "rotation vector" θ represents the incremental rotation of IMU relative to the inertial frame during the update period δt. To obtain θ Miller assumes that during the update period the outputs from the gyros can be described by a second order polynomial which requires two samples during the period. The following is the numerical solution for θ in which the very small triple vector product is neglected.

$$\theta = \delta_1 + \delta_2 + \tfrac{2}{3}(\delta_1 \times \delta_2)$$

where δ_1 and δ_2 are the incremental gyro samples taken at the mid-point and end-point of the update period respectively.

In order to keep processing time down when calculating the rotation quaternion $\overline{\theta}$ approximations are made for C and S. Miller proposes the use of either a 3rd order series expansion for sine and cosine, or a modified 2nd order expansion, claiming comparable performance. The 3rd order expansion has $C = 1 - \frac{1}{8}\theta \cdot \theta$, and $S = \frac{1}{2} - \frac{1}{48}\theta \cdot \theta$, while for the modified 2nd order expansion $C = 1 - \frac{1}{12}\theta \cdot \theta$ and $S = 0.5$.

When iterated at the fast rate the above method for calculating $\overline{Q}(t+\delta t)$ provides a representation for the attitude of the IMU over time in the inertial frame. Conversion of $\overline{Q}$ to be relative to an Earth based frame is carried out at the intermediate rate. The numerical equation to do this is

$$\overline{Q} \leftarrow \overline{\varphi}^{-1} * \overline{Q}$$

where $\overline{\varphi}^{-1} = C, -(\omega_{IE} t_I)S$ is the conjugate of the rotation quaternion $\overline{\varphi}$ for the rotation of the Earth base frame with respect to inertial space over the period of the intermediate rate t_I, and $\omega_{IE} t_I$ equals the "rotation vector" for the same. Note that the Earth base frame is usually defined on the surface of the Earth. Therefore ω_{IE} will be a function of latitude and thus needs to be updated periodically as the IMU moves across the Earth's surface. Miller does this at the intermediate rate.

Periodically it is necessary to correct scale errors (see Appendix 1) that accumulate in the quaternion. This is done at the intermediate rate by normalising the quaternion. In order once again to keep the computational time down Miller uses the following to normalise $\overline{Q}$.

$$QL = 1.5 - 0.5\sum_{i=0}^{3} Q_i^2$$

$$\overline{Q} \leftarrow QL.\overline{Q}$$

5.3 Linear Motion

Like attitude, linear motion can be split into motion caused by body external forces and those caused by gravitational effects, Earth rotation and curvature.

The velocity of the body, rotating with respect to an inertial frame, caused by external forces, is given by the equation

$$\dot{\mathbf{V}}_{IF} = \mathbf{F} - \Omega_{IB}\mathbf{V}_{IF}$$

where Ω_{IB} is the skew symmetric matrix for body rotation relative to the inertial frame and $\mathbf{F}$ is the sum of the external forces. In obtaining a solution to this equation for the fast rate calculations, Miller claims that it is a satisfactory compromise between computational load and accuracy to assume angular velocity and external forces are constant over the calculation period. The resulting numerical solution is

$$\mathbf{V}_{IF} \leftarrow \mathbf{V}_{IF} + \mathbf{A} - \theta \times (\mathbf{V}_{IF} + \tfrac{1}{2}\mathbf{A})$$

where $\mathbf{A} = \int \mathbf{F} dt$ is the accelerometer output, and θ is the "rotation vector". With $\mathbf{V}_{IF}$ set to zero at the start of an intermediate period, iteration at the fast rate gives the incremental change in velocity over the intermediate period.

For the velocity of body relative to the Earth, caused by gravitation and axes rotation over the intermediate period, the equation is

$$\dot{\mathbf{V}}_{EG} = \mathbf{g} - (2\Omega_{IE} + \Omega_{EN})\mathbf{V}_{EG}$$

where $\mathbf{g}$ is the gravity vector and Ω_{IE} and Ω_{EN} are the skew symmetric matrices for, respectively, the rotation of the Earth relative to the inertial frame, and the rotation of the navigation frame relative to the Earth frame. In this implementation the Earth and navigation frames are the same. The numerical solution to this is

$$\mathbf{V}_{EG} = \mathbf{g}t_i - (2\Omega_{IE} + \Omega_{EN})(\mathbf{V}_E + \tfrac{1}{2}\mathbf{g}t_i)t_i$$

where $\mathbf{V}_E \leftarrow \mathbf{V}_E + (\mathbf{V}_{IF} + \mathbf{V}_{EG})$ is the total velocity relative to the Earth calculated using $\mathbf{V}_{EG}$ from the previous intermediate calculations.

Position is updated using $\mathbf{X} \leftarrow \mathbf{X} + \{\mathbf{V}_E + \tfrac{1}{2}(\mathbf{V}_{IF} + \mathbf{V}_{EG})\}t_i$.

6. Software Details

The author has made several implementations of the algorithms discussed in chapter 5. These can be divided into two main groups, those that accept real time IMU data and those that take IMU data that has been previously stored. They have all been written in C++.

The following sections detail the various implementations. Text written in italics is explained further in Appendix 3.

6.1 IMU Sampling

The IMU (in this case a GIC100) provides accelerometer and gyro output at a rate of 600 Hz. The *fast rate calculation* period requires gyro samples at the mid point and end of the period and accelerometer samples at the end of the period. This means that the highest possible rate of calculations with the GIC100 is 300 Hz. In general 2n (n = 1, 2, 3...) *IMU Blocks* are needed for the *Fast Data Block*. For example, on a 33 MHz 386 PC it was found that a minimum of 4 IMU Blocks were needed for fast rate calculations in order to avoid computational overload, giving a maximum update rate of 150 Hz. With the greater computational over-heads of incorporating GPS data more IMU Blocks per fast rate calculations may be needed or a faster processor used.

6.2 Program Descriptions

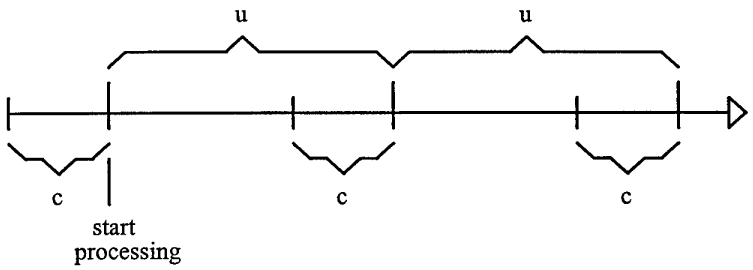
I_G_#???.CPP is the filename format for the main programs. # is either **R** for the program using real time input data, or **F** for input data read from a file. ??? refers to the particular version.

The following lists the source file names for the various implementations together with a brief description of the file's purpose. Support source files are also listed.

I_G_F1.CPP

This version only processes IMU data stored in a file. It features the following:

- Code that handles IMU data blocks that may or may not include a time stamp.
- Determination of the IMU biases, ie on the accelerometers and gyros before processing begins and again at regular periods during processing. Biases are calculated by simple averaging of IMU data over a period of time. This is implemented by the user supplying two parameters at run time. One for the period over which averaging takes place, and the other for the period at the end of which biases are updated. The following shows how this occurs during processing.



where u = the update period and c = the bias calculation period, both being user defined at run-time.

Note that with averaging it is essential the IMU is stationary during the bias calculation period. For this reason in dynamic tests recalculation is not practical. In these circumstances u is set to a value greater than the duration of the test.

In section 3 it was stated that the fast rate calculations require gyro samples to be taken at the middle and end of a Fast Data Block, and accelerometer samples at the end. A variation of this is implemented where for the gyros samples within the each half of the Fast Data Block are averaged and the accelerometer samples averaged over the entire block before being used in calculations. Assuming actual changes in rates and accelerations during a Fast Data Block are less than IMU noise then the signal to noise ratio will be improved. The impact of this becomes greater as the number of IMU blocks per Fast Data Block increases.

I_G_F2_A.CPP

This version processes both GPS and IMU data. All the features of version I_G_F1.CPP are contained in this version along with the following:

- The GPS data used is taken from a file containing output data from an Auspace Multinav receiver.
- The method of incorporating the GPS data is to simply update periodically the estimated position, velocity and attitude information being generated by the integration of IMU data. This is achieved by averaging GPS data for a period prior to the moment when updating occurs. INS position and velocity estimates are reset to those of the GPS average and the quaternion to represent a level orientation while maintaining INS estimated heading. It should be noted that by using an averaging technique for the GPS data that operation is limited to the static applications unless the averaging period is set to one second, in which case only one GPS sample is taken, thus allowing dynamic applications.
- It is possible to set an averaging period of the GPS data that is greater than the update period.

I_G_F2_T.CPP

This version is the same as I_G_F2_A.CPP with the only exception being the GPS data was taken from a Trimble receiver hence the GPS input data is in a different format.

I_G_F3.CPP.

This version is designed to test the algorithm's performance without the effects of inaccuracies in the input data from the IMU. Consequently simulated IMU data for known motion, eg coning, can be generated and stored in a file for use in this version. The following describe its differences from the version above.

- The IMU input data file has a different format of IMU data, using floating point numbers.
- No bias calculations are performed.
- No GPS data is used.

I_G_R1.CPP

This version processes real time data supplied from the GIC100 IMU only. It is a basic version with few added features. The following details its main features:

- IMU data is captured through the PC's DMA operation via a purpose built interface.
- Accelerometer and gyro biases are calculated by averaging data over a set period while the IMU is stationary and comparing them against accelerometer and gyro values calculated for the IMU's location and orientation. During this process the DMA handles one IMU Block of data at a time.
- For the integration process the fast rate calculations are performed only after there has been accumulated in a buffer via the DMA all the IMU data generated during an *intermediate rate calculation* period. When the buffer is full the DMA initiates an interrupt which in turn flags the commencement of fast rate calculations.

I_G_R2.CPP

This version differs from I_G_R1.CPP in one important way which is explained below

- As in I_G_R1.CPP two buffers are used in the integrating process, however their size is such that they can only store the data of one Fast Data Block each. When an interrupt is initiated after a buffer is full the fast rate calculations are carried out immediately while the other buffer is being filled. When enough fast rate calculations have been done an intermediate rate calculation is performed.

A drawback with this version is that with the smaller buffers than in the I_G_R1 version there is less time in which to perform extra calculations. Thus when GPS is introduced into the algorithm this method is more likely to require longer IMU sampling periods. On the other hand the advantage of this version is that the delay between the time the intermediate calculations are completed and the time for which they are valid will be less.

IMU.CPP

This file contains function definitions that are specific for the operation of the GIC100 IMU. The functions are only required by the real time main programs I_G_R???.CPP. The function declarations are made in the IMU.H file.

DMA.CPP

This file contains a set of functions to facilitate the operation of the PC's DMA. Some of these functions are required only by the real time main programs I_G_R???.CPP. The function declarations are made in the DMA.H file.

INTRPT.CPP

This file contains a set of functions to facilitate the operation of the PC's Interrupt ports. Some of these functions are required only by the real time main programs I_G_R???.CPP. The function declarations are made in the INTRPT.H file.

6.3 Main Program Structure

The following sections describe the structures of the various main program versions listed above.

6.3.1 Common Assumptions

For all the main programs versions listed above there are some common assumptions made, which are:

- The initial orientation of the IMU must be such that the Z-axis is pointing down.
- The IMU data is assumed to have been generated at 600 Hz.

6.3.2 Versions I_G_F1.CPP, I_G_F2_A, and I_G_F2_T

Figure 1 shows a block diagram that describes the structure of the main program of versions I_G_F1.CPP, I_G_F2_A.CPP and I_G_F2_T.CPP. The latter two differ from I_G_F1.CPP primarily by the code represented inside the dotted box after the function Output_Location(), in which GPS data is incorporated. Also for the latter two versions additional code has been made to the Initialise() function that opens the GPS input data file and reads from it into an array relevant data for the first GPS data averaging period.

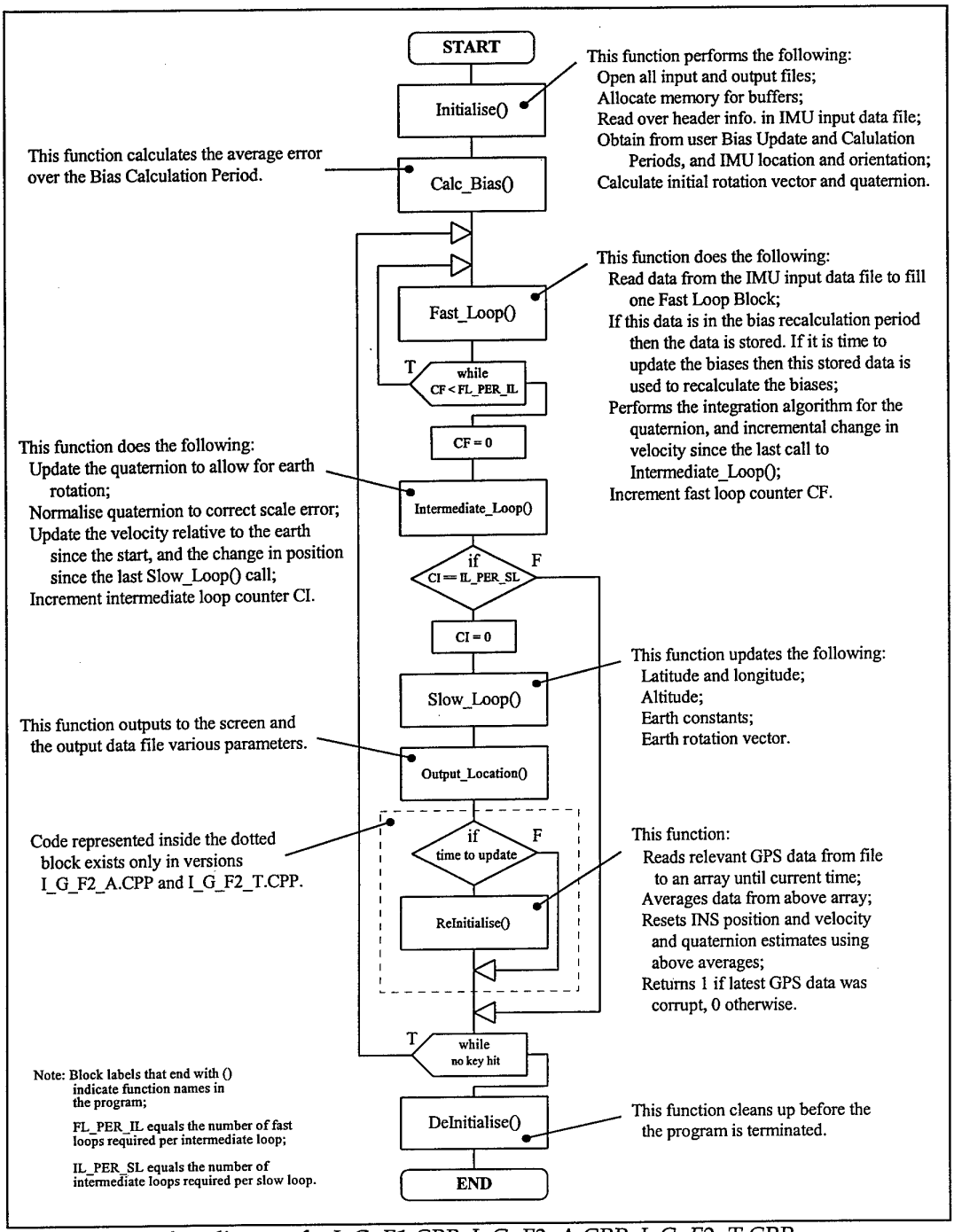


Figure 1. Flow diagram for I_G_F1.CPP, I_G_F2_A.CPP, I_G_F2_T.CPP.

In all three versions three files are opened:

IMU_IN.DAT

This is the binary input file containing data taken from a GIC100 IMU. These files are generated using a program called GIC100.EXE written by Greg Thamm. The file

consists of a series of data blocks each containing accelerometer and gyro values and an optional extra time stamp (see Appendix 3 under IMU Block). When the time stamp is included there is also some header information at the start of the file. The time information is never used, but the different format of files with or without time stamps has to be dealt with. This is done by defining *TIME_DATA_FLAG* as 1 in the include file I_G_FILE.H when the time stamp is used.

IMU_OUT.DAT

This is a binary output file that stores the accelerometer and gyro values (before conversion from GIC100 units) actually used in the fast rate calculations. Nine integer values are stored for each fast rate calculation. In order they are x, y, z accelerometer values, gyro values at mid-point of sample, and gyro values at end-point of sample.

INSGPS.DAT

This text output file stores various values during the execution of the Output_Location() function, that have been generated by the program. Eleven floating point numbers are written to file and in order are: Time (s) since the start of data processing (referenced to the input data, not real time); Displacement (m) from initial position in north, east, and altitude directions; Current velocity (m/s) relative to the Earth in north, east, and down directions; Quaternion scalar value Q_0, and vector values Q_1, Q_2, and Q_3.

For versions I_G_F2.A.CPP and I_G_F2_T.CPP an additional input file is opened for reading GPS data named GPS_IN.DAT. The Trimble and Auspace GPS receivers that generate the data stored in the file use completely different formats and provide more information than is required by these programs. In Appendix 3 the relevant data block formats are given under *Auspace Data* and *Trimble data.* In the Trimble data there is no velocity value for the vertical direction. This means that for version I_G_F2_T.CPP the vertical velocity estimate does not get altered when GPS updating takes place and thus remains solely the product of IMU integration.

The following user input requirements are the same for all three versions.

- Bias update period.
- Bias calculation period.
- Initial latitude, longitude, and altitude.
- Initial heading

For the versions including GPS data the period when GPS data is used to update the position, velocity and quaternion must also be supplied by the user.

6.3.3 Version I_G_F3.CPP

Figure 2 shows the general structure of version I_G_F3.CPP. Comparing to figure 1 it can be seen that structurally it differs from version I_G_F1.CPP by only the removal of the function that calculates the IMU biases.

Additional differences not shown in the figures are as follows.

- As pointed out in section 6.2 this version uses floating point numbers in the IMU input data file (which is named TEST_IN.DAT). The format of this data is shown in Appendix 3 under *IMU Test Block.*
- An additional input data file named I_G_INIT.DAT is used to read in some of the data that the user would otherwise be required to enter on execution. The file is a text file and the data stored in it, in order, is the initial latitude, longitude, altitude, and the heading. This data is placed all on one line and separated by "#" symbols. For all bar altitude there are two values separated by a space for degrees and minutes with the latter being a decimal number if necessary. For example:
- -34 42.603# 138 38.435# 25# 0 0#
- With no biasing or GPS data, and I_G_INIT.DAT being used, this version does not require any data to be entered by the user during execution.

6.3.4 Version I_G_R1.CPP and I_G_R2.CPP

Figure 3 shows the structure of main section and its interrupt functions of I_G_R1.CPP, while the same for version I_G_R2.CPP is shown in figure 4. Some functions relating to the interrupt and DMA for the IMU have not been described specifically. These functions are defined in IMU.CPP, DMA.CPP, and INTRPT.CPP where further details of their operation are given.

For both versions there are many common aspects. These are dealt with first and the unique aspects are described directly after the applicable figures.

There are two output files generated. These are called IMU_OUT.DAT and INSGPS.DAT and are identical to the output files of the same name described in section 6.3.2.

The only user input required during execution is the initial location of the IMU, ie latitude longitude and altitude, and the heading.

The IMU interface is connected to DMA channel 7 and Interrupt channel 10. The basic operational relationship between the IMU interface, DMA, and Interrupt is as follows.

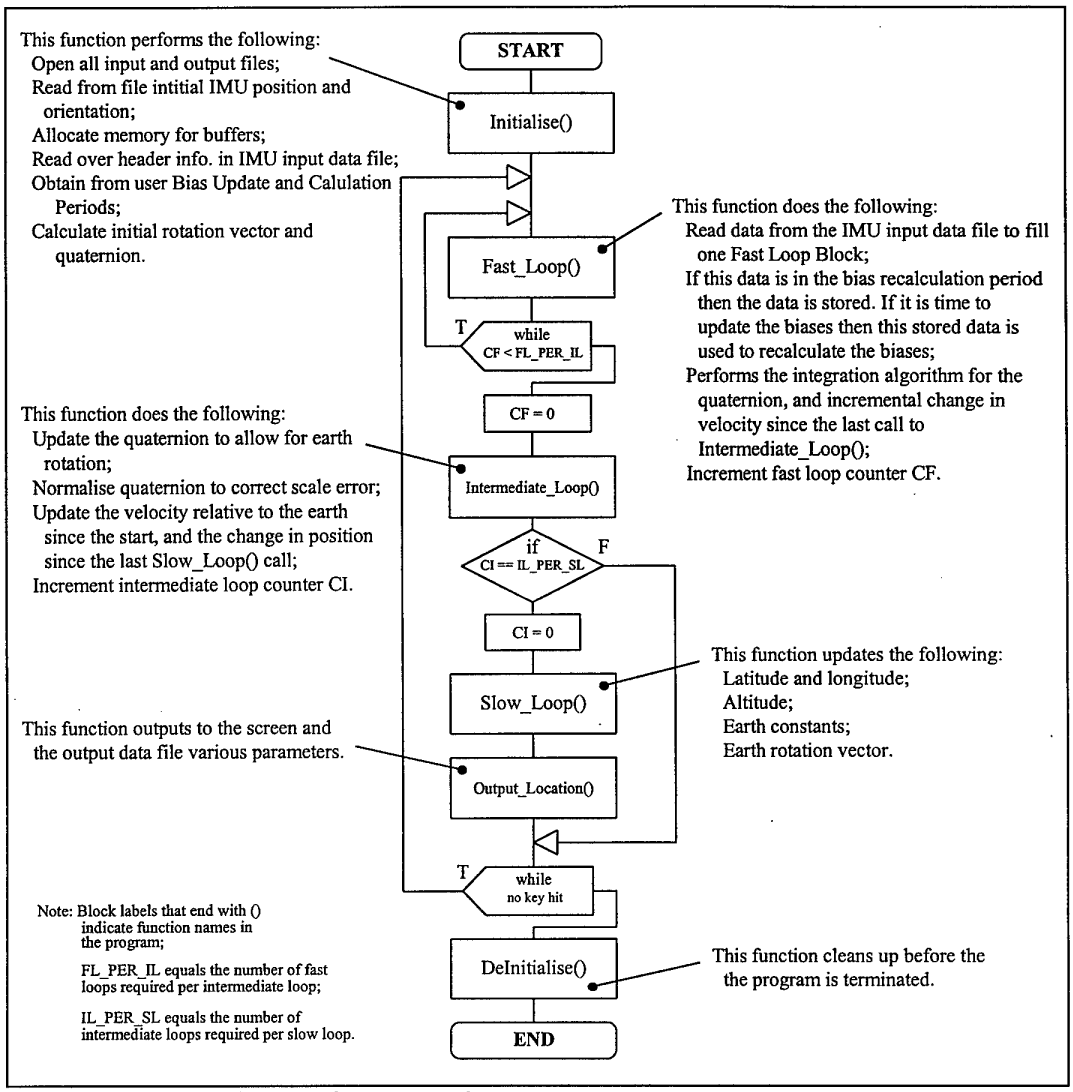


Figure 2. Flow diagram for I_G_F3.CPP

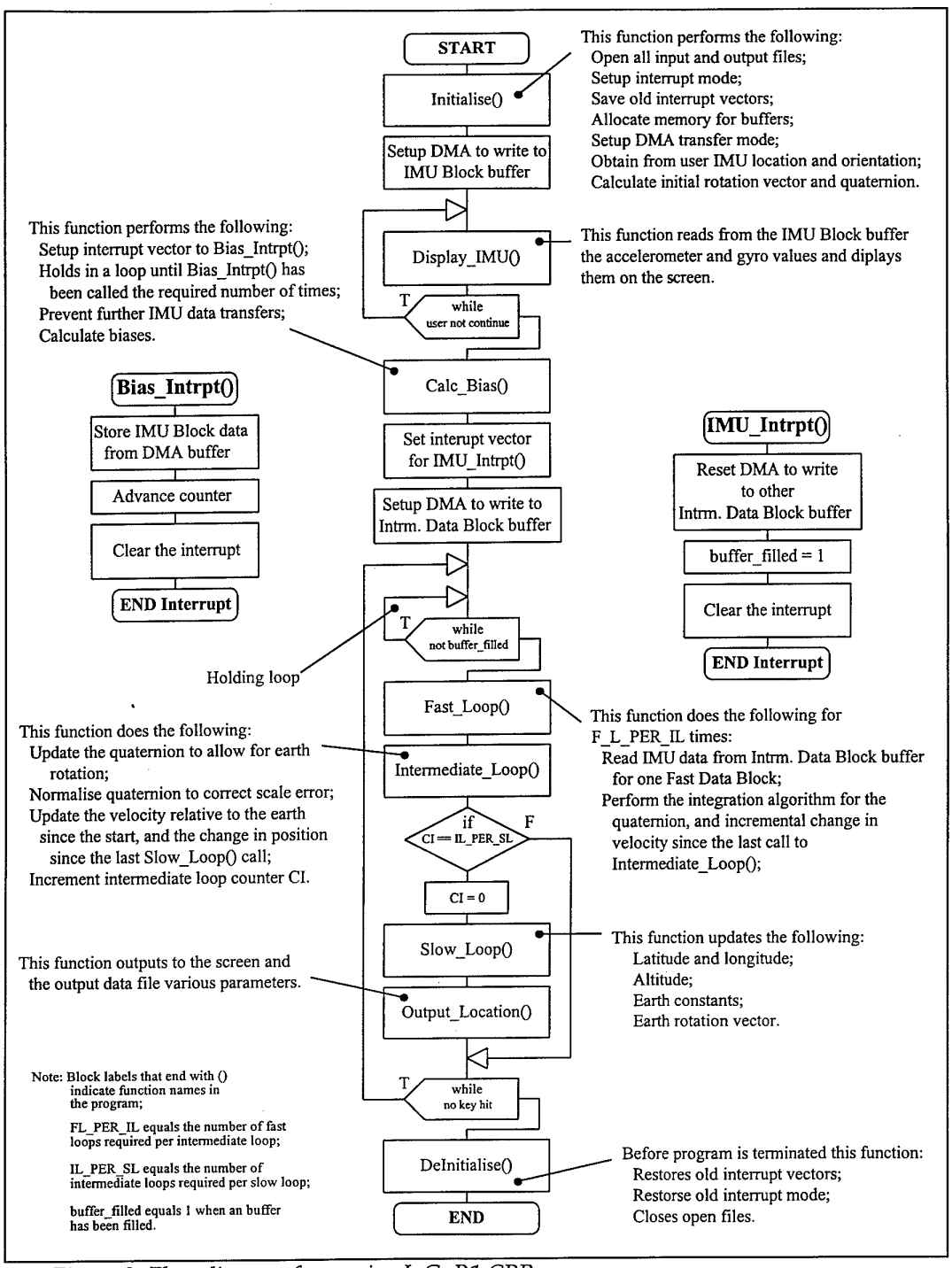


Figure 3. Flow diagram for version I_G_R1.CPP.

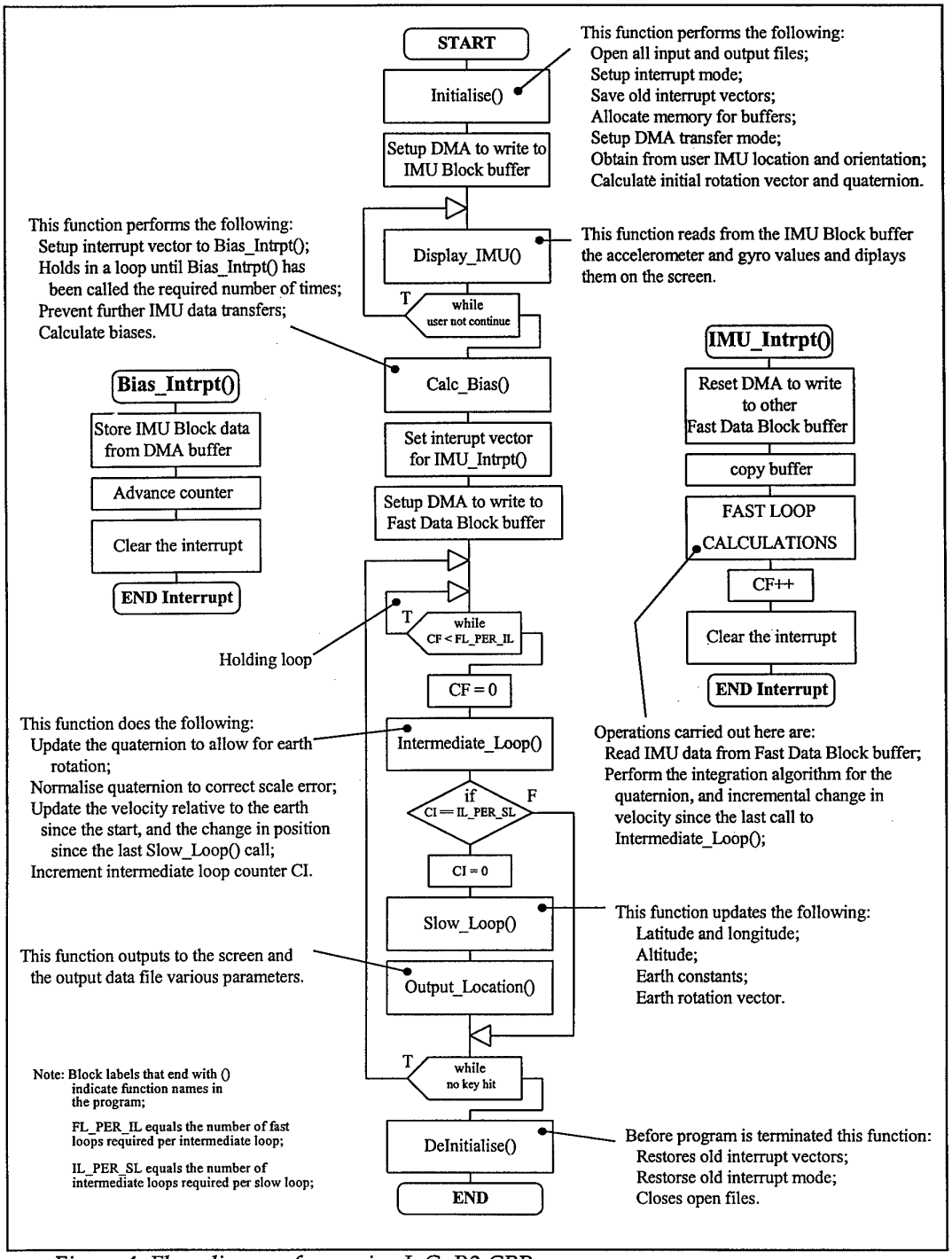


Figure 4. Flow diagram for version I_G_R2.CPP.

DMA channel 7 is set up to write transfer, autoinitialise, and single transfer modes while the Interrupt controller is set to Special Fully Nested Mode (for more details refer to Section 3 and 5 of reference 2). This is achieved through the function calls Set_DMA_Mode() and Initialise_IC() respectively. Before transfers take place DMA

channel 7 is given an address of a buffer to transfer to and the number of words to transfer through the function call Load_IMU_DMA(), and Interrupt channel 10 has its vector set to an interrupt function through a call to Set_Interrupt_Vector(). With this done transfers can now take place. When the interface has IMU data ready to send it notifies the DMA to commence transfer. Once the DMA has counted the required number of transfers it reinitialises itself and notifies the IMU interface. The Interface responds by requesting an interrupt so that the interrupt function can then operate on the data stored in the buffer. The process then is repeated.

There are two different sized buffers, one is for the calculation of biases while the other is for the actual integration process. Likewise there are two separate interrupt functions.

For the bias calculations the buffers size is such that it will take one IMU Block of data. The name of the interrupt function is Bias_Intrpt(). The duration in seconds over which biases are calculated, and hence the number of times Bias_Intrpt() is called, is determined by *BIAS_TIME* which is defined in I_G_REAL.H.

For the integration process in version I_G_F1.CPP two separate buffers are allocated, each capable of storing an Intermediate Data Block. The program sets up the DMA to alternate between the buffers as each is filled. When a buffer is full IMU_Intrpt() is requested to swap DMA buffers. On return from IMU_Intrpt() the program drops out of a holding loop to commence integration calculations. First the fast rate calculations are repeated until all the data in the most recently filled buffer has been used. Next one set of intermediate rate calculations are performed and finally if sufficient intermediate rate calculations have already occurred, a slow rate calculation is performed. When these have all been completed the program returns to the holding loop to await the next IMU_Intrpt() request. While processing is taking place on the data in one buffer, the other is being filled. Data will be lost if IMU_Intrpt() is requested before the completion of all the fast rate calculations on the data in the previously filled buffer.

For the integration process in version I_G_R2.CPP two separate buffers are allocated, each capable of storing a Fast Data Block. The program sets up the DMA to alternate between the buffer as each is filled. When a buffer is full IMU_Intrpt() is requested. This function swaps the DMA buffers and then performs one set of fast rate calculations. When IMU_Intrpt() has finished the program returns to a holding loop. After the required number of iterations of IMU_Intrpt() have occurred the program falls out of the holding loop to do the intermediate rate calculations and if sufficient intermediate rate calculations have already occurred, the slow rate calculations are performed. The program must complete all this and return to the holding loop before the next IMU_Intrpt() is requested otherwise data will be lost.

6.4 Unresolved Problems

In the real time versions a problem was encountered in which occasionally the program would lock up the computer. It appeared that critical memory was being overwritten, probably by the DMA. Investigation found that there are delays of random frequency and duration during processing and when the delays are long enough the calculations are not complete on a particular buffer's data when an

interrupt request to change buffers occurs. The problem can be avoided by increasing the number of IMU Blocks per set of fast rate calculations and/or the number of fast rate calculation sets per intermediate rate calculation set. In effect this provides more processing time for the calculations thus accommodating the delays as part of the processing time. This is not an entirely desirable solution since the slower processing rate is counter productive. Investigations continue.

6.5 Planned Modifications

The following functional additions and or changes are planned, which will result in new main program versions that either source their data in real time or from file.

- A new calibration routine for initialising the IMU that will use Kalman Filtering techniques.
- Inclusion of integrity checks that will monitor the following for unexpected results:

  1. The sensor data;
  2. The global solution ;
  3. Algorithms for the fusion of data, ie Kalman Filters.

  Bad results in any of the above will lead to the rejection of the contributing data.

7. References

1. R. B. Miller, "Strapdown inertial navigation systems: An algorithm for attitude and navigation computations", ARL-SYS-REPORT-23, October 1980.
2. Intel 82350 EISA Chip Set Manual, August 1990.

Appendix 1
Quaternion Definition and Properties

A quaternion uses four parameters to define the rotation of one 3 dimensional coordinate system relative to another. Miller defines it as:

$$\overline{Q} = C, \theta S$$

where: $C = \cos(\frac{1}{2}\theta_0)$; $S = (1/\theta_0)\sin(\frac{1}{2}\theta_0)$; $\theta = (\theta_1, \theta_2, \theta_3)$ is the "rotation vector" (assuming $\theta_1, \theta_2, \theta_3$ are small); and $\theta_0 = (\theta_1^2 + \theta_2^2 + \theta_3^2)^{\frac{1}{2}}$.

The physical interpretation of this is that of a rotation through an angle θ_0 measured from reference to body axes, about a unit vector defined (in both axes systems) by θ/θ_0.

Normalisation of Quaternions

With the quaternion defined as above the sum of the squares of the four parameters is unity. The square root of this is often called the "***length***" of the quaternion.

Should the length change from unity then a "***scale error***" has taken place. This can be corrected by "*normalisation*", ie by dividing the quaternion by its length.

Quaternion "Multiplication"

Given the quaternions $\overline{A} = A_0, \mathbf{A}$ and $\overline{B} = B_0, \mathbf{B}$, then their product $\overline{C} = C_0, \mathbf{C}$ is

$$\overline{C} = \overline{A} * \overline{B} = (A_0 B_0 - \mathbf{A} \cdot \mathbf{B}), \{A_0 \mathbf{B} + \mathbf{A} B_0 + (\mathbf{A} \times \mathbf{B})\}.$$

To physically interpret this consider a body where $\overline{A}$ represents the rotation from reference to body axes. If the body is rotated relative to the reference frame so that $\overline{B}$ represents the rotation from the old to new body axes, then the quaternion $\overline{C}$ represents the rotation from the reference to the new body axes.

Refer to Appendix 2 of Miller for further details.

Appendix 2
3rd Order Taylor Series and 4th Order Runge-Kutta Solutions

The following are alternative solutions to $\dot{\overline{Q}}(t) = \frac{1}{2}\overline{Q}(t) * \omega(t)$ to find $\overline{Q}(t+h)$.

3rd Order Taylor Series:

$$\overline{Q}(t+h) = \overline{Q}(t) * \overline{U}(t)$$

where: $\overline{U} = U_0, \mathbf{U}$ is a quaternion defined by;

$U_0 = 1 - \frac{1}{8}(\delta \cdot \delta)$;

$\mathbf{U} = \frac{1}{2}\delta + \frac{1}{3}(\delta_1 + \delta_2) - \frac{1}{48}(\delta \cdot \delta)\delta$;

h = the sample period;

δ_1 is the integrated gyro sample from t to $(t + \frac{1}{2}h)$;

δ_2 is the integrated gyro sample from $(t + \frac{1}{2}h)$ to $(t+h)$;

$\delta = \delta_1 + \delta_2$;

4th Order Runge-Kutta:

With δ_1 and δ_2 defined as above let

$$\mathbf{H}\omega_0 = (3\delta_1 - \delta_2) \text{ and } \mathbf{H}\omega_1 = (\delta_1 + \delta_2) \text{ and } \mathbf{H}\omega_2 = (3\delta_2 - \delta_1)$$

then

$$\overline{k_0} = \frac{1}{2}\overline{Q}(t) * \overline{H\omega_0}$$

$$\overline{Q_1}(t + \tfrac{1}{2}h) = \overline{Q}(t) + \tfrac{1}{2}\overline{k_0}$$

$$\overline{k_1} = \frac{1}{2}\overline{Q_1}(t + \tfrac{1}{2}h) * \overline{H\omega_1}$$

$$\overline{Q_2}(t + \tfrac{1}{2}h) = \overline{Q}(t) + \tfrac{1}{2}\overline{k_1}$$

$$\overline{k_2} = \frac{1}{2}\overline{Q_2}(t + \tfrac{1}{2}h) * \overline{H\omega_1}$$

$$\overline{Q_3}(t + h) = \overline{Q}(t) + \overline{k_2}$$

$$\overline{k_3} = \frac{1}{2}\overline{Q_3}(t + h) * \overline{H\omega_2}$$

and finally

$$\overline{Q}(t+h) = \overline{Q}(t) + \tfrac{1}{6}(\overline{k_0} + 2(\overline{k_1} + \overline{k_2}) + \overline{k_3})$$

Appendix 3
Terminology

fast rate calculations: Where integration of IMU data is performed in body axes. Performed at highest frequency (100-300 Hz).

intermediate rate calculations: Corrects quaternion for Earth rotation, converts to Navigation coordinates and integrates to location. Frequency is 1-10 Hz.

slow rate calculations: Updates Earth constants. Frequency s 0.1-1 Hz.

FL_PER_IL: integer. Number of times fast rate calculations are performed before intermediate rate calculations are initiated.

IL_PER_SL: integer. Number of times intermediate rate calculations are performed before slow rate calculations are initiated.

CF: A counter used to count the number of times fast rate calculations have been performed since intermediate rate calculations were performed last.

CI: A counter used to count the number of times intermediate rate calculations have been performed since slow rate calculations were performed last.

IMU Block: A single block of data from the IMU, 24 bytes long (optional 32). Contains gyro and accelerometer info for 3 axes plus an optional time stamp. The IMU Block detail for the GIC100 is as follows:

word 1	X accel	word 2	X accel tag = x000d
3	Z accel	4	Z accel tag = x0009
5	Y accel	6	Y accel tag = x000b
7	Y gyro	8	Y gyro tag = x000e
9	Z gyro	10	Z gyro tag = x000a
11	X gyro	12	X gyro tag = x000c
13	option time	14	optional time

One word consists of a 2 byte integer.

To convert accelerometer values to m/s/s they must be multiplied by 1.1963e-1, and the gyros must be multiplied by 1.28e-3 to convert them to rad/s.

Fast Data Block: A block of data containing 2 n (n = 1, 2, 3,...) IMU Blocks which is the minimum required to perform a single fast rate calculation.

Intermediate Data Block: A block of data containing *FL_PER_IL* number of *Fast Data Blocks*. This is the amount of data that must be processed before an Intermediate rate calculations will be performed.

IMU Test Block: A single block of data containing 3 accelerometer values and 3 gyro values. The values have been generated by a program that calculates the exact accelerometer and gyro values for a defined motion. The format of the IMU Test Block is as follows:

word 1	X accel
2	Y accel
3	Z accel
4	X Gyro
5	Y Gyro
6	Z Gyro

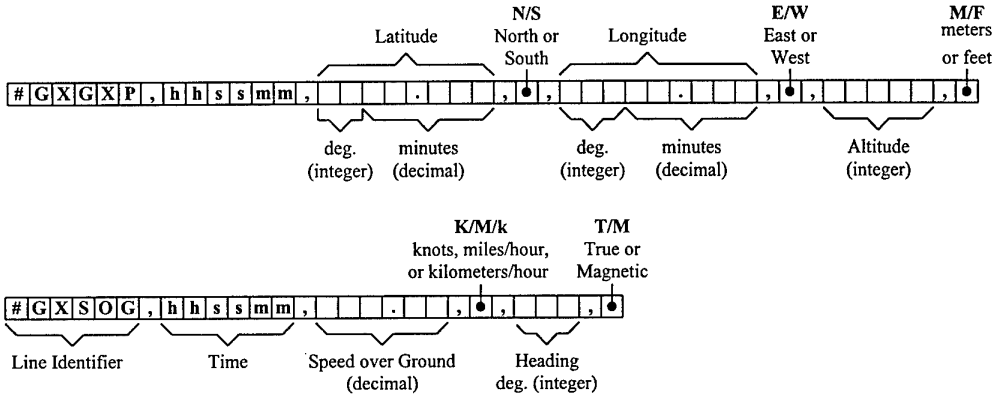
One word consists of an 8 byte double floating point.

The values are already in m/s/s and rad/s for accelerometers and gyros respectively.

Auspace Data: The relevant data required by version I_G_F2_A.CPP is contained in "message 100". This message is in a binary form. Its format is as follows.

Byte No.	Name	Type	Data
1	Start Byte	1 Byte	02 hex
2	Header	1 Byte	CC hex
3 - 4	Receiver ID	2 Byte Integer	000 - 999 decimal
5 - 6	Time Stamp, Week No.	2 Byte Integer	Standard GPS Week Number
7 - 10	Time Stamp, TOW	4 Byte Float	Seconds
11 - 15	Satellite Numbers	5 x 1 Byte Integers	Satellites used by KF
16 - 19	Position, Latitude	4 Byte Float	Radians, WGS 84
20 - 23	Position, Longitude	4 Byte Float	Radians, WGS 84
24 - 27	Position, Altitude	4 Byte Float	Radians, WGS 84
28 - 31	Velocity, North	4 Byte Float	Meters/Sec
32 - 35	Velocity, East	4 Byte Float	Meters/Sec
36 - 39	Velocity, Up	4 Byte Float	Meters/Sec
40	CheckSum	1 Byte Integer	Sum of bytes 3 to 39
41	End Byte	1 Byte	03 hex

Trimble Data: The relevant data required by version I_G_F2_T.CPP is contained in two lines of text, with a constant number of characters, generated by the Trimble receiver. These are as follows:



buffer_filled: A flag used in version I_G_R1.CPP that is set to 1 by an interrupt function when an Intermediate Data Block has been passed to memory via the DMA controller.

TIME_DATA_FLAG: Flag in the file data version of IMU.H (see sect. 4.1) that when equal to 1 allows the handling of *IMU blocks* that include a time stamp at the end.

DISTRIBUTION LIST

Low Cost INS-GPS Integration for Navigation (INS Software Implementation)
A General Document.

N. Luckman

DEFENCE ORGANISATION

Defence Science and Technology Organisation

Chief Defence Scientist
FAS Science Policy
AS Science Industry Interaction
AS Science Corporate Management
} shared copy

Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington
Scientific Adviser to Thailand MRD (Doc Data Sheet only)
Senior Defence Scientific Adviser/ScientificAdviser Policy and Command (shared copy)
Navy Scientific Adviser
Scientific Adviser - Army
Air Force Scientific Adviser
Director Trials

Electronics and Surveillance Research Laboratory

Director (1 copy)

Aeronautical and Maritime Research Laboratory

Director
Chief, Weapons Systems Division and Research Leaders shared copy
HGC
Dave Hards, Guidance & Control Group
Ashley Martin, Guidance & Control Group
Sanjay Mazumdar, Guidance & Control Group
Nick Luckman (author), Guidance & Control Group

DSTO Library

Library Fishermens Bend
Library Maribyrnong
Main Library DSTOS (2 copies)
Library, MOD, Pyrmont (Doc Data Sheet only)

Defence Central

OIC TRS, Defence Central Library
Officer in Charge, Document Exchange Centre (DEC), 1 copy
DEC requires the following copies of public release reports to meet exchange agreements under their management:
*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Centre, 2 copies
*Canada Defence Scientific Information Service
*NZ Defence Information Centre
National Library of Australia

Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet only)

Air Force
Director General Force Development (Air)

Army
Director General Force Development (Land)
ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)
SO (Science), HQ 1 Division, Milpo, Enoggera, Qld 4057 (Doc Data Sheet)
NAPOC QWG Engineer NBCD c/- DENGRS-A, HQ Engineer Centre Liverpool Military Area, NSW 2174 (Doc Data Sheet)

Navy
Director General Force Development (Sea),
Director Combat Force Development (Sea),
Director Capability Development Analysis (Sea),
SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000.

UNIVERSITIES AND COLLEGES
Australian Defence Force Academy Library
Senior Librarian, Hargrave Library, Monash University

OTHER ORGANISATIONS
NASA (Canberra)
AGPS

ABSTRACTING AND INFORMATION ORGANISATIONS
INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
American Society for Metals
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS
Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (10 copies)

Total 60 copies

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
Commercial-in-Confidence

2. TITLE

Low Cost INS-GPS Integration for Navigation (INS Software Implementation)

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document (U)
Title (U)
Abstract (U)

4. AUTHOR(S)

N. Luckman

5. CORPORATE AUTHOR

Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Vic 3001

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-GD-0081	AR-009-692	General Document	March 1996

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
J 9505-8-26	ADA93/276	ADFA	30	2

13. DOWNGRADING/DELIMITING INSTRUCTIONS

To be reviewed March 1999.

14. RELEASE AUTHORITY

Chief, Weapons Systems Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Global Positioning System
Inertial Navigation
Inertial measurement units
Algorithms
Computer Programs

19. ABSTRACT

This document describes several versions of software that have been developed to implement an Inertial Navigation System algorithm. The form of the algorithm is intended to facilitate the combining of Global Positioning System data into the INS solution, and a simple example (position and velocity reset) is included in the software.